Rocky Enterprise Linux 9.2 Manual Pages on command 'systemd-cryptenroll.1'

$ man systemd-cryptenroll.1

SYSTEMD-CRYPTENROLL(1) systemd-cryptenroll SYSTEMD-CRYPTENROLL(1)

NAME

systemd-cryptenroll - Enroll PKCS#11, FIDO2, TPM2 token/devices to LUKS2 encrypted volumes

SYNOPSIS

systemd-cryptenroll [OPTIONS...] [DEVICE]

DESCRIPTION

systemd-cryptenroll is a tool for enrolling hardware security tokens and devices into a LUKS2 encrypted volume, which may then be used to unlock the volume during boot. Specifically, it supports tokens and credentials of the following kind to be enrolled:

1. PKCS#11 security tokens and smartcards that may carry an RSA key pair (e.g. various YubiKeys)
2. FIDO2 security tokens that implement the "hmac-secret" extension (most FIDO2 keys, including YubiKeys)
3. TPM2 security devices
4. Regular passphrases
5. Recovery keys. These are similar to regular passphrases, however

are randomly generated on the computer and thus generally have higher entropy than user-chosen passphrases. Their character set has been designed to ensure they are easy to type in, while having high entropy. They may also be scanned off screen using QR codes. Recovery keys may be used for unlocking LUKS2 volumes wherever passphrases are accepted. They are intended to be used in combination with an enrolled hardware security token, as a recovery option when the token is lost.

In addition, the tool may be used to enumerate currently enrolled security tokens and wipe a subset of them. The latter may be combined with the enrollment operation of a new security token, in order to update or replace enrollments.

The tool supports only LUKS2 volumes, as it stores token meta-information in the LUKS2 JSON token area, which is not available in other encryption formats.

LIMITATIONS

Note that currently when enrolling a new key of one of the five supported types listed above, it is required to first provide a passphrase or recovery key (i.e. one of the latter two key types). For example, it's currently not possible to unlock a device with a FIDO2 key in order to enroll a new FIDO2 key. Instead, in order to enroll a new FIDO2 key, it is necessary to provide an already enrolled regular passphrase or recovery key. Thus, if in future key roll-over is desired it's generally recommended to combine TPM2, FIDO2, PKCS#11 key enrollment with enrolling a regular passphrase or recovery key.

Also note that support for enrolling multiple FIDO2 tokens is currently not too useful, as while unlocking systemd-cryptsetup cannot identify which token is currently plugged in and thus does not know which authentication request to send to the device. This limitation does not apply to tokens enrolled via PKCS#11 ? because tokens of this type may be identified immediately, before authentication.

OPTIONS

The following options are understood:

--password

Enroll a regular password/passphrase. This command is mostly equivalent to cryptsetup luksAddKey, however may be combined with --wipe-slot= in one call, see below.

--recovery-key

Enroll a recovery key. Recovery keys are mostly identical to passphrases, but are computer-generated instead of being chosen by a human, and thus have a guaranteed high entropy. The key uses a character set that is easy to type in, and may be scanned off screen via a QR code.

--unlock-key-file=PATH

Use a file instead of a password/passphrase read from stdin to unlock the volume. Expects the PATH to the file containing your key to unlock the volume. Currently there is nothing like --key-file-offset= or --key-file-size= so this file has to only contain the full key.

--pkcs11-token-uri=URI

Enroll a PKCS#11 security token or smartcard (e.g. a YubiKey). Expects a PKCS#11 smartcard URI referring to the token. Alternatively the special value "auto" may be specified, in order to automatically determine the URI of a currently plugged in security token (of which there must be exactly one). The special value "list" may be used to enumerate all suitable PKCS#11 tokens currently plugged in. The security token must contain an RSA key pair which is used to encrypt the randomly generated key that is used to unlock the LUKS2 volume. The encrypted key is then stored in the LUKS2 JSON token header area.

In order to unlock a LUKS2 volume with an enrolled PKCS#11 security token, specify the pkcs11-uri= option in the respective /etc/crypttab line:

```
myvolume /dev/sda1 - pkcs11-uri=auto
```

See crypttab(5) for a more comprehensive example of a systemd-cryptenroll invocation and its matching /etc/crypttab line.

--fido2-credential-algorithm=STRING

Specify COSE algorithm used in credential generation. The default value is "es256". Supported values are "es256", "rs256" and "eddsa".

"es256" denotes ECDSA over NIST P-256 with SHA-256. "rs256" denotes 2048-bit RSA with PKCS#1.5 padding and SHA-256. "eddsa" denotes EDDSA over Curve25519 with SHA-512.

Note that your authenticator may not support some algorithms.

--fido2-device=PATH

Enroll a FIDO2 security token that implements the "hmac-secret" extension (e.g. a YubiKey). Expects a hidraw device referring to the FIDO2 device (e.g. /dev/hidraw1). Alternatively the special value "auto" may be specified, in order to automatically determine the device node of a currently plugged in security token (of which there must be exactly one). The special value "list" may be used to enumerate all suitable FIDO2 tokens currently plugged in. Note that many hardware security tokens that implement FIDO2 also implement the older PKCS#11 standard. Typically FIDO2 is preferable, given it's simpler to use and more modern.

In order to unlock a LUKS2 volume with an enrolled FIDO2 security token, specify the fido2-device= option in the respective /etc/crypttab line:

```
myvolume /dev/sda1 - fido2-device=auto
```

See crypttab(5) for a more comprehensive example of a systemd-cryptenroll invocation and its matching /etc/crypttab line.

--fido2-with-client-pin=BOOL

When enrolling a FIDO2 security token, controls whether to require the user to enter a PIN when unlocking the volume (the FIDO2 "clientPin" feature). Defaults to "yes". (Note: this setting is without effect if the security token does not support the "clientPin" feature at all, or does not allow enabling or disabling it.)

--fido2-with-user-presence=BOOL

When enrolling a FIDO2 security token, controls whether to require the user to verify presence (tap the token, the FIDO2 "up" feature) when unlocking the volume. Defaults to "yes". (Note: this setting is without effect if the security token does not support the "up" feature at all, or does not allow enabling or disabling it.)

--fido2-with-user-verification=BOOL

When enrolling a FIDO2 security token, controls whether to require user verification when unlocking the volume (the FIDO2 "uv" feature). Defaults to "no". (Note: this setting is without effect if the security token does not support the "uv" feature at all, or does not allow enabling or disabling it.)

--tpm2-device=PATH

Enroll a TPM2 security chip. Expects a device node path referring to the TPM2 chip (e.g. /dev/tpmrm0). Alternatively the special value "auto" may be specified, in order to automatically determine the device node of a currently discovered TPM2 device (of which there must be exactly one). The special value "list" may be used to enumerate all suitable TPM2 devices currently discovered.

In order to unlock a LUKS2 volume with an enrolled TPM2 security chip, specify the tpm2-device= option in the respective /etc/crypttab line:

```
myvolume /dev/sda1 - tpm2-device=auto
```

See crypttab(5) for a more comprehensive example of a systemd-cryptenroll invocation and its matching /etc/crypttab line.

Use --tpm2-pcrs= (see below) to configure which TPM2 PCR indexes to bind the enrollment to.

--tpm2-pcrs= [PCR...]

Configures the TPM2 PCRs (Platform Configuration Registers) to bind the enrollment requested via --tpm2-device= to. Takes a "+" separated list of numeric PCR indexes in the range 0...23. If not used, defaults to PCR 7 only. If an empty string is specified, binds the enrollment to no PCRs at all. PCRs allow binding the enrollment to specific software versions and system state, so that

the enrolled unlocking key is only accessible (may be "unsealed")

if specific trusted software and/or configuration is used.

Table 1. Well-known PCR Definitions

```
????????????????????????????????????
?PCR ? Explanation                 ?
????????????????????????????????????
?0   ? Core system firmware       ?
?    ? executable code; changes   ?
?    ? on firmware updates        ?
????????????????????????????????????
?1   ? Core system firmware       ?
?    ? data/host platform         ?
?    ? configuration; typically   ?
?    ? contains serial and model  ?
?    ? numbers, changes on basic  ?
?    ? hardware/CPU/RAM           ?
?    ? replacements               ?
????????????????????????????????????
?2   ? Extended or pluggable      ?
?    ? executable code; includes  ?
?    ? option ROMs on pluggable   ?
?    ? hardware                   ?
????????????????????????????????????
?3   ? Extended or pluggable      ?
?    ? firmware data; includes    ?
?    ? information about          ?
?    ? pluggable hardware         ?
????????????????????????????????????
?4   ? Boot loader and additional ?
?    ? drivers; changes on boot   ?
?    ? loader updates. The shim   ?
?    ? project will measure the   ?
?    ? PE binary it chain loads   ?
```

? ? into this PCR. If the ?

? ? Linux kernel is invoked as ?

? ? UEFI PE binary, it is ?

? ? measured here, too. sd- ?

? ? stub(7) measures system ?

? ? extension images read from ?

? ? the ESP here too (see ?

? ? systemd-sysext(8)). ?

????????????????????????????????????

?5 ? GPT/Partition table; ?

? ? changes when the ?

? ? partitions are added, ?

? ? modified or removed ?

????????????????????????????????????

?6 ? Power state events; ?

? ? changes on system ?

? ? suspend/sleep ?

????????????????????????????????????

?7 ? Secure boot state; changes ?

? ? when UEFI SecureBoot mode ?

? ? is enabled/disabled, or ?

? ? firmware certificates (PK, ?

? ? KEK, db, dbx, ...) ?

? ? changes. The shim project ?

? ? will measure most of its ?

? ? (non-MOK) certificates and ?

? ? SBAT data into this PCR. ?

????????????????????????????????????

?9 ? The Linux kernel measures ?

? ? all initrds it receives ?

? ? into this PCR. ?

????????????????????????????????????

?10 ? The IMA project measures ?

? ? its runtime state into ?

? ? this PCR. ?

??????????????????????????????????

?11 ? systemd-stub(7) measures ?

? ? the ELF kernel image, ?

? ? embedded initrd and other ?

? ? payload of the PE image it ?

? ? is placed in into this ?

? ? PCR. Unlike PCR 4 (where ?

? ? the same data should be ?

? ? measured into), this PCR ?

? ? value should be easy to ?

? ? pre-calculate, as this ?

? ? only contains static parts ?

? ? of the PE binary. Use this ?

? ? PCR to bind TPM policies ?

? ? to a specific kernel ?

? ? image, possibly with an ?

? ? embedded initrd. systemd- ?

? ? pcrphase.service(8) ?

? ? measures boot phase ?

? ? strings into this PCR at ?

? ? various milestones of the ?

? ? boot process. ?

??????????????????????????????????

?12 ? systemd-boot(7) measures ?

? ? any specified kernel ?

? ? command line into this ?

? ? PCR. systemd-stub(7) ?

? ? measures any manually ?

? ? specified kernel command ?

? ? line (i.e. a kernel ?

? ? command line that ?

? ? overrides the one embedded ?

? ? in the unified PE image) ?

? ? and loaded credentials ?

? ? into this PCR. (Note that ?

? ? if systemd-boot and ?

? ? systemd-stub are used in ?

? ? combination the command ?

? ? line might be measured ?

? ? twice!) ?

??????????????????????????????????

?13 ? systemd-stub(7) measures ?

? ? any systemd-sysext(8) ?

? ? images it loads and passed ?

? ? to the booted kernel into ?

? ? this PCR. ?

??????????????????????????????????

?14 ? The shim project measures ?

? ? its "MOK" certificates and ?

? ? hashes into this PCR. ?

??????????????????????????????????

For most applications it should be sufficient to bind against PCR 7 (and possibly PCR 14, if shim/MOK is desired), as this includes measurements of the trusted certificates (and possibly hashes) that are used to validate all components of the boot process up to and including the OS kernel. In order to simplify firmware and OS version updates it's typically not advisable to include PCRs such as 0 and 2 in the binding of the enrollment, since the program code they cover should already be protected indirectly through the certificates measured into PCR 7. Validation through these certificates is typically preferable over validation through direct measurements as it is less brittle in context of OS/firmware updates: the measurements will change on every update, but code signatures likely will validate against pre-existing certificates.

--tpm2-with-pin=BOOL

When enrolling a TPM2 device, controls whether to require the user to enter a PIN when unlocking the volume in addition to PCR binding, based on TPM2 policy authentication. Defaults to "no". Despite being called PIN, any character can be used, not just numbers.

Note that incorrect PIN entry when unlocking increments the TPM dictionary attack lockout mechanism, and may lock out users for a prolonged time, depending on its configuration. The lockout mechanism is a global property of the TPM, systemd-cryptenroll does not control or configure the lockout mechanism. You may use tpm2-tss tools to inspect or configure the dictionary attack lockout, with tpm2_getcap(1) and tpm2_dictionarylockout(1) commands, respectively.

--tpm2-public-key= [PATH], --tpm2-public-key-pcrs= [PCR...], --tpm2-signature= [PATH]

Configures a TPM2 signed PCR policy to bind encryption to. The --tpm2-public-key= option accepts a path to a PEM encoded RSA public key, to bind the encryption to. If this is not specified explicitly, but a file tpm2-pcr-public-key.pem exists in one of the directories /etc/systemd/, /run/systemd/, /usr/lib/systemd/ (searched in this order), it is automatically used. The --tpm2-public-key-pcrs= option takes a list of TPM2 PCR indexes to bind to (same syntax as --tpm2-pcrs= described above). If not specified defaults to 11 (i.e. this binds the policy to any unified kernel image for which a PCR signature can be provided).

Note the difference between --tpm2-pcrs= and --tpm2-public-key-pcrs=: the former binds decryption to the current, specific PCR values; the latter binds decryption to any set of PCR values for which a signature by the specified public key can be provided. The latter is hence more useful in scenarios where software updates shell be possible without losing access to all previously encrypted LUKS2 volumes.

The --tpm2-signature= option takes a path to a TPM2 PCR signature file as generated by the systemd-measure(1) tool. If this this is not specified explicitly a suitable signature file tpm2-pcr-signature.json is searched for in /etc/systemd/, /run/systemd/, /usr/lib/systemd/ (in this order) and used. If a signature file is specified or found it is used to verify if the volume can be unlocked with it given the current PCR state, before the new slot is written to disk. This is intended as safety net to ensure that access to a volume is not lost if a public key is enrolled for which no valid signature for the current PCR state is available. If the supplied signature does not unlock the current PCR state and public key combination, no slot is enrolled and the operation will fail. If no signature file is specified or found no such safety verification is done.

--wipe-slot= [SLOT...]

Wipes one or more LUKS2 key slots. Takes a comma separated list of numeric slot indexes, or the special strings "all" (for wiping all key slots), "empty" (for wiping all key slots that are unlocked by an empty passphrase), "password" (for wiping all key slots that are unlocked by a traditional passphrase), "recovery" (for wiping all key slots that are unlocked by a recovery key), "pkcs11" (for wiping all key slots that are unlocked by a PKCS#11 token), "fido2" (for wiping all key slots that are unlocked by a FIDO2 token), "tpm2" (for wiping all key slots that are unlocked by a TPM2 chip), or any combination of these strings or numeric indexes, in which case all slots matching either are wiped. As safety precaution an operation that wipes all slots without exception (so that the volume cannot be unlocked at all anymore, unless the volume key is known) is refused.

This switch may be used alone, in which case only the requested wipe operation is executed. It may also be used in combination with any of the enrollment options listed above, in which case the enrollment is completed first, and only when successful the wipe

operation executed ? and the newly added slot is always excluded from the wiping. Combining enrollment and slot wiping may thus be used to update existing enrollments:

systemd-cryptenroll /dev/sda1 --wipe-slot=tpm2 --tpm2-device=auto

The above command will enroll the TPM2 chip, and then wipe all previously created TPM2 enrollments on the LUKS2 volume, leaving only the newly created one. Combining wiping and enrollment may also be used to replace enrollments of different types, for example for changing from a PKCS#11 enrollment to a FIDO2 one:

systemd-cryptenroll /dev/sda1 --wipe-slot=pkcs11 --fido2-device=auto

Or for replacing an enrolled empty password by TPM2:

systemd-cryptenroll /dev/sda1 --wipe-slot=empty --tpm2-device=auto

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

SEE ALSO

systemd(1), systemd-cryptsetup@.service(8), crypttab(5), cryptsetup(8), systemd-measure(1)